Incentives for Mobile Cloud Environments through P2P Auctions

Ismael Cuadrado-Cordero*, Anne-Cécile Orgerie†, Christine Morin*

*Inria, Rennes, France,
Email: {ismael.cuadrado-cordero,christine.morin}@inria.fr
†CNRS, IRISA, France
Email: anne-cecile.orgerie@irisa.fr

***Abstract*—Mobile Cloud Computing is a form of collaborative decentralized Cloud which allows mobile devices to unload computation to a local Cloud formed by mobile and static devices. Mobile Cloud Computing provides a better service to latency sensitive applications, due to its physical proximity to the VM host. However, in these systems, the problem of free riding users becomes more acute, for the heterogeneity of devices (from smartphones to private servers) makes the gap of contributed resources much larger. In this work, we analyze the use of incentives for Mobile Clouds, and propose a new auction system adapted to the high dynamism and heterogeneity of these systems. We compare our solution to other existing auctions systems in a Mobile Cloud use case, and show the suitability of our solution.**

I. Introduction

Cloud computing has become a dominant solution for prompt and on-demand leasing of computing resources. In this technology, the architecture of public cloud datacenters where data are centralized is commonly deployed. However, many alternatives to the datacenter-centralized vision of the Cloud exist in literature. This is the case of Social Clouds. The definition of a Social Cloud (SC) - "a resource and service sharing framework utilizing relationships established between members of a social network" - was firstly proposed by [3]. In these networks the decentralization of resources follows an overall pattern of interest. This makes SC especially relevant in nowadays ever-growing Internet. given that the system should be able to absorb the amount of computation and data generated in the system and users' devices most of the time. An implementation of SC is found on Mobile Computing (MC). These are infrastructures composed of both static and mobile devices, born to address the limitations of mobile devices in terms of resources and connectivity [6]. MC improves resource-hungry mobile services, by offloading data and computation into the Cloud [6].

However, decentralized clouds may be unfair when some users contribute more resources than others. These users, called free riders, are estimated to account for the largest percentage of users in collaborative networks [9]. The unfairness associated with the free riding problem represent an obstacle to the adoption of a collaborative technology. In our problem, to incentivize the use of MCs it is necessary to extend the concept of lease. A lease is a contractual arrangement between an entity, which rent part of its computational power, and a group of users, which offers a payment in return. In a MC lease the entity offering the computational power is formed by a group of users, called sellers. The sellers rent part of their resources to host the Virtual Machines (VMs) used to provide the service. On the other hand, the rest of users, called buyers, pay the sellers for hosting the VM. As a consequence to this new concept of lease, a pricing system is required.

Existing solutions are mainly based on two leasing models: fixed and negotiated pricing. In a fixed price system, a seller offers its resources at a specific cost, and the buyers match it. In a negotiated price system, the price of the resource is established by direct competition (auction) between buyers and sellers. In this work, we propose a multi-sided auction system, where the user becomes both buyer and seller, auctioning on other users as needed. Furthermore, we propose an open auction system where the application provider supervises the process, and has the possibility of bidding along with one or more users if the expected result of the auction is unfair to other users.

In this paper we target interactive MC applications, with multiple active users. In those, the VM hosting the application is placed in a device which suits every user using it. However, the hosting device may not make use of the application or be willing to share its resources. To solve this issue, an incentives system is necessary for the users to be more willing to lending resources. This incentive is recorded in a lease contract, which is signed between the host and the users, fitting all actors involved. To our knowledge, no other work has researched on the use of incentives in the case of MCs up to the moment of writing this paper.

The remaining of the paper is as follows. Section II shows the background and a classification on existing solutions. In Section III, the use case scenario is introduced. Our solution is explained in Section IV, while the result from our experimentation is shown on V. Finally, Section VI draws our main conclusions and future work.

II. Background and Motivation

Decentralized systems have set lately the focus of research on Mobile Clouds (MCs), due to properties like low latency to the user, reduced energy consumption and robustness [6]. MCs are infrastructures composed of both static and mobile devices, such as PCs and smartphones, but also domestic servers and components from smart-cities' infrastructures such as routers or specific purpose hardware. These clouds eliminate the need of a datacenter, allocating the computation in the user

and/or network devices. They provide a very small latency and high energy efficiency [3], [5], due to the small geographical distance between users.

We review existing lease-provision systems and investigated their adequacy to MC. We identified three types of leases:

- **1 to 1 (Standard lease provision):** The conditions of the lease are established beforehand through a Service-Level Agreement (SLA). Through a SLA, a device commits to lending a part of its resources to provide a negotiated service. Examples of this approach can be found in IaaS providers such as Amazon AWS [2] or Google Cloud [8]. This solution is, however, too rigid for dynamic environments, due to issues such as predefined slots of time or dependence on the SLA offered by the seller, which is caused by lack of competition.

- **1 to N (Simple Auction):** Simple auctions start with a SLA over which the price is negotiated. They are either direct or reverse. In the *direct* approach (one seller, multiple buyers) the SLA is offered beforehand by the seller and the buyer offering the highest price gets it. In the *reverse* one (one buyer, multiple sellers) the SLA is offered beforehand by the buyer and the seller requesting the smallest price to fulfill it gains the auction. In a direct 1 to N auction, the seller remains unchanged and values a lease according to the maximum amount payable for it, while in a reverse one an unchanged buyer values it to the minimum pay. A known example of a direct 1 to N approach is Amazon's EC2 Spot Instances [1] and existing works on MC such as [10], [11]. This type of lease is less rigid than the standard one. However, these approaches do not encourage competition either between buyers or sellers.

- **N to N (Double Auction):** Double auctions encourage competition both in-between multiple buyers and sellers. It can be described as a combination of direct and reverse auctions, where the SLA and price are decided according to the market offer [13], [15], [16]. Double auctions are the most adaptable solution to the MC context, because the conditions are always set dynamically based on the offer and demand.

All leases, independently from its type, are unfair when one of the buyers possesses more monetary resources than the rest, unbalancing the market. To face this circumstance, several buyers may join efforts to gain an auction against more wealthy opponents, called a group auction. A specific kind of group auctions are group-buying auctions, in which a special price is promised to buyers under the condition of reaching a minimum bid [14]. Bids are either informed (buyers know about the value given to the service by other participants) or blind (no information about other bids are known to buyers). Group auctions are of special importance for the targeted applications, where multiple buyers collaborate on the same instance of the application.

III. Scenario: Neighborhood Applications on Mobile Clouds

We have chosen to study incentives to the use of neighborhood interactive applications deployed on a MC architecture, such as described in [4]. In this scenario multiple users collaborate using applications in a MC restricted to a neighborhood. That is, users are all located inside the same neighborhood and use services applications adapted to this neighborhood, as in real-life applications such as [7]. We also assume a smart-city infrastructure in the neighborhood. This infrastructure provides data about the area of use to the neighbors. Every node inside this neighborhood is a candidate to be used as host for a VM (including network and smart-city equipment and mobile devices). We understand by node any device working in the network, which an application layer, such as computers, smartphones, domestic or ISP routers or specific purpose hardware for smartcities infrastructures.

For each instance of a service running in the neighborhood, an overlay network called microcloud is created, as defined in [5]. Different roles are dynamically assigned to nodes:

- **Service Manager (SM)**, which controls the life-cycle of the application. It acts as the auctioning authority.
- **Service Provider (SP)**, which hosts the application backend in the form of a VM.
- **Client**, which makes use of the service. Both SM and SP can be at the same time clients of the microcloud.

Additionally, two super nodes are assigned to the smart-city infrastructure. This roles are Base Provider, which backs-up data, and the Base Manager, for the management of the overall infrastructure (management of microclouds). The backing-up process is done through a snapshot of the VM. The copies of the VM are never accessed by clients, and do not interfere with the system. Thus, there still exist no replication of active data.

Confining the traffic to the neighborhood, this architecture eliminates replication (since only one copy of the data is active at a given moment, hosted in the SP). Also, it reduces the amount of data traveling across the network (as a consequence of the elimination of the replication and the utilization of efficient connections); and consumes less energy and provides a better Quality of Experience (QoE) than in an architecture where the VM is hosted in a datacenter.

The characteristics of this scenario are:

Highly dynamic: The mobility of the different mobile nodes produce a highly dynamic system. Thus, auctions are started on demand by buyers (clients).

Client orientation: The system needs to take into account different users requirements (such as CPUs, RAM, storage space or expected QoE). Every client might request different conditions for the SLAs.

Minimum satisfaction: End-users' QoE is a main concern in the design of the system, as interactive applications are considered. Thus, a minimum satisfaction needs to be guaranteed to nodes.

Credit mobility: Users' credit can be used to pay for different applications in different microclouds. If users move between neighborhoods, their credit moves along with them.

IV. Our approach: P2P Auction with External Supervision

As described in the previous section, both SP and SM (unique per microcloud) consume more resources than the rest of nodes, and they need to be compensated accordingly. Thus, an auction system is required as an incentive. We propose a multi-sided auction system between peered clients (P2P). When the service is launched for the first time the SM is assigned among the nodes through a 1 to N reverse auction by the Base Manager. Once assigned, the SM assigns the SP. To select the new SP, every node blind bids on one or more nodes. Finally, the node with the highest bid obtains the SP role. The use of blind auctioning increases trust from the seller being paid fairly, given that the buyer's perception is not affected by other bids. As the clients move in the network, the SP may be reallocated when one or more clients decide that, under the current topology, the location of the SP is not satisfactory enough for them, and request to start a new auction.

In our approach, a common SLA is proposed to the clients, and each node gives certain importance to those characteristics that it considers important, creating a personalized SLA. To ease the auction, sellers expose information about their capacities (information useful for the SLA). These resources include, but are not limited to, CPUs, RAM, Probability of Failure or Energy Efficiency. We define the satisfaction of a node as the difference between what it required and what it gets, similar to other works such as [13], [15]. As an example, we show Equation 3. Here, the SLA is considered as a linear relation between the trust that the community has on the seller, the fulfillment of the buyer (difference between what it requests and what the seller provides) and the distance between the buyer and seller (in ms).

$$SLA = Trust * Fulfillment - Distance$$

$$\begin{aligned} Fulfillment = \alpha * (reqPoF - PoF) + \beta * (EE - reqEE) \\ +\gamma * (RAM - reqRAM) + \delta * (CPUs - reqCPUs) \end{aligned}$$

$$Distance = \epsilon * \sum (lat_n, i) \quad (1)$$

Trust represents the confidence the system has that the seller is honest about its resources, and is set between 0 and 1. This evaluation is made by users, according to the historical of the service of this specific node. *reqPoF, reqEE,reqRAM* and *reqCPUs* represent, respectively, the Probability of Failure - probability of the system failing based on historic, between 0 and 1-, Energy Efficiency - difference between energy consumed while being a client and hosting a service in the historic -, RAM and CPUs requested by the node. *PoF, EE, RAM* and *CPUs* represent the Probability of Failure, Energy Efficiency, RAM and CPUs offered by the node to bid on. α, β, γ, and δ define the relative weight of each variable. lat_n, i represents latency between nodes *n* and *i* and ϵ the weight of the distance in the overall SLA. $\alpha, \beta, \gamma, \delta$, and ϵ are configurable by the node, to better fit the users' requirements. This equation can be extended depending on the users' requirements. As stated before, the existence of a trust evaluation is necessary to increase confidence on the seller. If a node does not comply with the accepted SLA (which is detected by the rest of clients), then it will be stripped of the lease without being paid and its confidence variable is reduced.

To reduce the possibility of sellers providing fake information about themselves, a recommendation system is implemented, where the history of sellers' performance is evaluated publicly. We extend the idea proposed by [11] through a voting system, where users can evaluate the service received. A node with a poor historical performance will receive a negative feedback, reducing its possibilities to host new leases. Finally, the existence of a central entity (the SM) ensures that in the case of an irregularity (such as a misbehavior from the node or connectivity problems) the lease is canceled and the node hosting the service is taken the VM away and sanctioned, while the auction process is relaunched. In order to ensure the QoE of the clients, the SM is allowed to participate in the auction - using credit provided by the application provider -, aligning with one or more clients if one of the clients abuses a powerful position over the rest.

The auction process remains unchanged but, after the auction, the satisfaction of the users is evaluated and, if it is found to be unsatisfactory for one or more users the SM is contacted. While the market value of the lease remains the same as it was decided through the auction, the SM may group with other buyers and buy the a lease which satisfies better all the buyers. While this solution restricts the freedom of the auction process, it ensures that there is not a node or a group of buyers which unfairly condition the QoE of the rest of users. The QoE is not only necessary for users, but also for the service owner, which improves the perception users have of its service. This process is described in Algorithm 1.

V. Evaluation

The aim of our experiments is to evaluate the performance of different auction mechanisms in a microcloud-based platform operating within a neighborhood, and its effect on a users' incentive scheme. To do so, we focus on users' satisfaction and rewarding. Experiments have been run using NS3 [12], a packet level simulator. We used a network of 45 static devices, shown in Figure 1, over which a set of mobile devices (between 50 and 100) randomly move. The devices simulated in the experiments are heterogeneous, with different demands and offers of resources. On this infrastructure, a 45-minutes trace of a real shared on-line document has been reproduced. This trace has been produced for the experimentation, given the lack of traces on multiple-users interactive Cloud applications fitting our requirements in literature. This trace includes several users writing and deleting on an on-line document sharing system. 15 different auctions occur during the experiments, resulting from the mobile devices' mobility.

On this network we use different bidding strategies for allocating the SP. For each experiment, all devices in the network (both static and mobile devices) start with an initial credit of 10 units. When the VM is deployed, the Service

Algorithm 1 Auction's External Supervision.

for all $nodes$ **do**
  Initialize node.bid to 0
for each $node$ in $nodes$ **do**
  for each $neighbor$ of $node$ **do**
    {Bid only on those $nodes$ matching its requirements}
    if $neighbor$ matches $node$ requirements **then**
      node.bid(neighbor, minimumBid
      +(satisfaction)*biddingMoney/2)
Initialize $currentBid$ to $minimumBid$
Initialize $currentAvgSatisfaction$ to $-\infty$
for each $node$ in $nodes$ **do**
  if $node.bid$ > $currentBid$ **then**
    if $currentAvgSatisfaction$ >= $threshold$ AND $node.AvgSatisfaction$ < $threshold$ **then**
      $currentBid$ = $node.bid$ - $currentBid$
    else if $currentAvgSatisfaction$ < $threshold$ AND $currentAvgSatisfaction$ < $node.AvgSatisfaction$ **then**
      $currentAvgSatisfaction$=$node.AvgSatisfaction$
      $chosenNode$ = $node$
      $currentBid$ = $node.bid$
$chosenNode.receiveBid$
return $chosenNode$

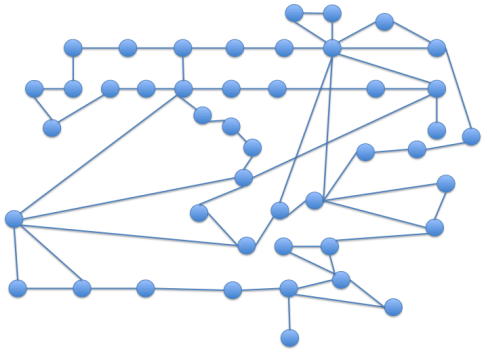


Fig. 1: Topology of the network of static nodes

Manager automatically chooses the node which fits the needed requirements and requires the least credit. Over time, the mobile devices move along the network, and request a re-allocation of the VM which satisfies them better. In our experiments, to ease the comparison of results, we used the SLA described before in Equation 3, and set the same values for $\alpha = 10, \beta = 10, \gamma = 1, \delta = 1$ and $\epsilon = 1$ on each node. Using constants for the users' chosen variables simplifies the comparison of the behavior of the different approaches based on the mobility of nodes and heterogeneousness of devices.

The bidding formula used to determine the amount of money to bid on is described on Equation 2.

$$bid = minimumBid + (satisfaction) * biddingMoney/\sigma \quad (2)$$

minimumBid and σ are set by the user. *minimumBid* determines what is the minimum amount to bid and σ the aggressiveness on which the node bids (how fast it increases the amount of money to bid). In our experiments, a *minimumBid* value, in a range between 0 to 10 credits and different for each node, is set for each node. The value $\sigma = 2$ has been used for all clients, to evaluate the results over similar conditions. The bidding strategies used are:

Standard Double Auction (SDA): Each node bids on its direct neighbors a sum relative to its satisfaction with that node. This strategy rewards low latency and is the most aggressive, as each node only accepts its maximum satisfaction.

Flexible Double Auction (FDA): It is similar to SDA but besides bidding on its direct neighbors, it also considers distance. To ease comparisons, delay has been considered equal between every connection, so instead of in ms, distance has been counted on hops. Thus, each node bids a smaller percentage of what was bid in the previous layer of clients on clients with a distance greater than 1 and lesser than 10. FDA is less aggressive and less focused on latency than SDA.

Collaborative Double Auction (CDA): As in FDA, in this strategy every node bids on its direct neighbors and those with a distance of more than 1, but every group - set of clients which share the node to be bid on as a direct neighbor - proposes a common bid. A node may be part of different groups. While every node bids less credit, their total contribution is still significant. It rewards credit saving for future bids.

Standard Double Auction with External Supervision (SDA-ES): Similar to SDA, but the SM has the option of matching the biggest bid to provide a better average satisfaction in the system. The interference of the SM only happens when the average satisfaction is lesser than a threshold. This strategy ensures that no wealthy buyer controls the outcome of the auction in detriment of the community (that is, forcing a low satisfaction on the rest).

Random assignment (RND): Assigns the VM randomly. Is used as a base line for the comparison.

For the current experiments we have set a threshold of 5 units, so the average satisfaction of clients is, if possible, bigger than this. This value has been chosen after experimental consideration of different candidates, shown in Figure 2. For our experimentation, five different values where considered: $threshold = 0units$, $threshold = 5units$, $threshold = -5units$, $threshold = 10units$ and $threshold = 15units$. Experiments where performed using a variable number of participants, under the same conditions and network as the rest of experimentation.

As it is shown in Figure 2, the use of a threshold set to 5 units performs well in all the situations. Also, it allows us showing that the addition of external credits in the auction produces, sometimes, feedbacks in the system. For example, for 30 and 40 clients and $threshold = 5units$, the average satisfaction remains well over the expected minimum. This is caused because, every time an auction is corrected, some extra credits are pushed into the system, redistributing the overall wealth in the system.

After, we evaluated the clients' satisfaction under the different bidding strategies, shown on Figure 3 for a variable number of mobile clients. Average, maximum and minimum data are also displayed in Tables I, II and III.

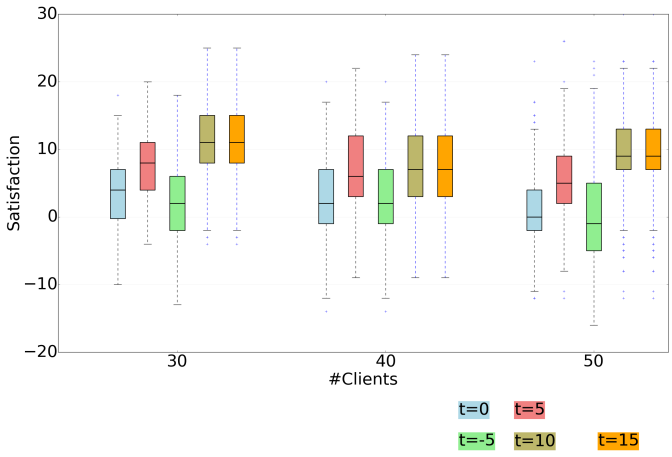


Fig. 2: Client satisfaction using different threshold values

TABLE I: Average satisfaction per strategy

Strat. \ #Clients	50	60	70	80	90	100
SDA	1.62	5.70	-5.39	1.62	-1.31	2.53
FDA	1.62	6.49	-4.37	2.53	-1.31	3.55
CDA	0.15	-0.18	0.60	-0.29	-5.50	-3.46
SDA-ES	12.61	8.76	5.70	7.74	2.76	5.81
RND	0.72	0.49	-2.44	-1.31	-3.4	1.51

As depicted, while there is not a great difference between using an SDA or FDA auction, collaborative auctions do not perform well in this scenario - getting, in most cases, a satisfaction close to the one of the random assignment. While CDA is less aggressive and more fair between sellers, having every user with roughly equal credit, the probabilities of a winning bid which does not satisfy the set are bigger. However, under certain circumstances, it may show an equal or better performance than the two others. SDA-ES performs always better than the rest, due to the external supervision.

TABLE II: Maximum satisfaction per strategy

Strat. \ #Clients	50	60	70	80	90	100
SDA	19.74	20.65	10.68	20.76	19.63	22.80
FDA	19.74	22.69	10.68	20.76	19.63	22.80
CDA	17.82	22.80	16.69	20.76	12.72	22.58
SDA-ES	25.86	26.88	20.65	26.77	19.63	23.82
RND	20.65	17.71	18.39	20.65	19.63	21.67

TABLE III: Minimum satisfaction per strategy

Strat. \ #Clients	50	60	70	80	90	100
SDA	-14.40	-13.20	-21.36	-18.30	-17.28	-15.36
FDA	-16.38	-13.20	-21.36	-18.30	-17.28	-15.36
CDA	-23.17	-25.44	-22.26	-22.38	-24.30	-26.34
SDA-ES	-4.37	-2.2	-9.2	-3.40	-9.24	-7.2
RND	-16.42	-17.28	-21.36	-19.31	-22.38	-24.30

TABLE IV: Maximum bid per strategy

Strategy \ #Clients	50	60	70	80	90	100
SDA	151	120	230	241	321	341
FDA	160	120	230	260	321	341
CDA	160	120	230	260	321	341
SDA-ES	540	190	300	260	550	350

The choice of a threshold of satisfaction in a SDA-ES strategy is of paramount importance. Choosing an unrealistic - too high - satisfaction threshold that no node meets affects the freedom of credit exchange, as the SM will always decide. On the other hand, a very low threshold is always met and the strategy is the same as SDA. Furthermore, modifying this threshold, for example, providing a minimum satisfaction rather than an average one has undesired effects. Due to some users being ”unsatisfiable”, the global satisfaction is diminished to the same level as CDA.

The distribution of credit among clients participating in the auction at the end of the experiments is shown in Figure 4a. Here, every block represents the amount of money that a device have by the end of the experiment. While the distribution of credit among clients is more balanced using a CDA strategy, it also shows that by using a SDA strategy the total amount of credit used in the network is smaller than in any other. This is caused by the nature of the strategy, in which the number of participants in the auction is smaller, as shown in Figure 4b. This situation is caused by the unfair distribution of credit, which leads to a small number of clients affecting the result and reducing the number of clients involved in the auction. For the other strategies, we have not observed any difference in the number of clients involved. Last, the SDA-ES strategy shows a better distribution of credit than SDA and FDA, making the system more fair. While CDA provides a more equal distribution of credit than SDA-ES, it has also been shown that the global satisfaction tends to be much worse. As expected, the case of Random Assignment is not shown in Figure 4, given that no exchange of money is necessary.

However, using an SDA-ES strategy also increases the total amount of credit in the system - thus, for the users -, through external injections. This situation is shown in Table IV, where the biggest bid for each strategy is shown. As shown, having more credit in the system leads the buyers to offer bigger bids.

VI. Conclusions and Future Work

The expansion of geographically located MCs is a milestone in returning the cloud to users. Furthermore, as the mobile technologies advance and become more accessible, it will be the best option both to absorb the needed increase in the demand of energy, reduce the unnecessary network traffic and to provide a near to real-time experience. However, to be accepted by users, Mobile Clouds need to be incentivized from the application providers.

We believe that the use of a credit-based system enhances the perception of the MC as a local infrastructure, while the commerce of credit between users in exchange of services adapts the service to the neighborhood. On the other hand, it allows users to go beyond the purely digital domain by

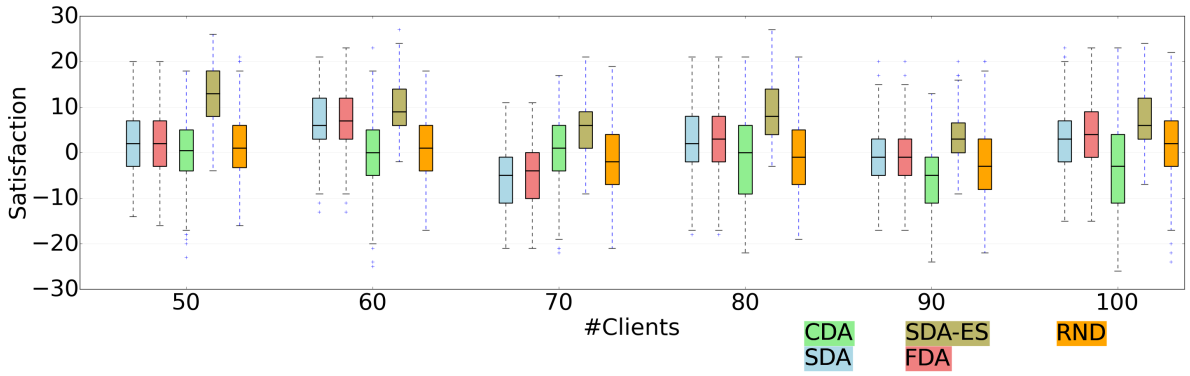
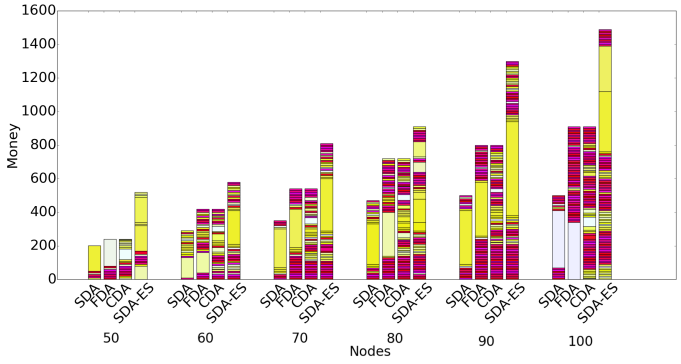
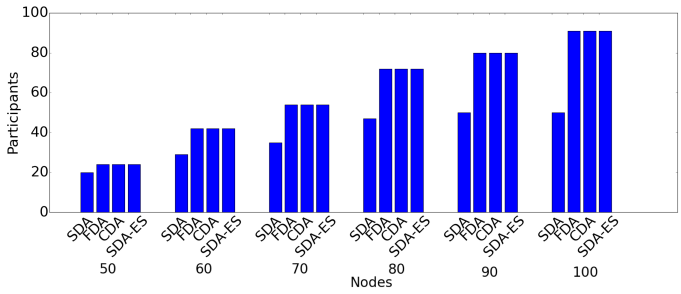


Fig. 3: Client satisfaction using different strategies



(a) Credit distribution among clients using different strategies



(b) Participants in the auction using different strategies

Fig. 4: Credit distribution vs. participants

exchanging this credit by real-life services. It also enhances the adaptability of services to neighborhoods, since larger quantities of credit in a neighborhood increases the number of services and, thus, the credit distribution.

Also, the company obtains first hand geographically located information. This information is of basic need in today's world of big data and pushes application providers into investing in the community which enhances users' experience.

References

[1] O. Agmon Ben-Yehuda, M. Ben-Yehuda, A. Schuster, and D. Tsafrir, "Deconstructing Amazon EC2 Spot Instance Pricing," in *IEEE Int. Conf. on Cloud Computing Technology and Science (CloudCom)*, 2011, pp. 304–311.

[2] AmazonTM, "Amazon cloud infrastructure," http://aws.amazon.com/about-aws/global-infrastructure/, 2016.

[3] K. Chard, K. Bubendorfer, S. Caton, and O. Rana, "Social Cloud Computing: A Vision for Socially Motivated Resource Sharing," *IEEE Transactions on Services Computing*, vol. 5, no. 4, pp. 551–563, 2012.

[4] I. Cuadrado-Cordero, F. Cuadrado, C. Phillips, A.-C. Orgerie, and C. Morin, "Microcities: a Platform based on Microclouds for Neighborhood Services," Inria, Research Report RR-8885, Feb. 2016. [Online]. Available: https://hal.inria.fr/hal-01291536

[5] I. Cuadrado Cordero, A.-C. Orgerie, and C. A. Morin, "GRaNADA: A Network-Aware and Energy-Efficient PaaS Cloud Architecture," in *IEEE International Conference on Green Computing and Communications (GreenCom)*, Sydney, Australia, Dec. 2015. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01205905

[6] N. Fernando, S. W. Loke, and W. Rahayu, "Mobile cloud computing: A survey," *Future Generation Computer Systems*, vol. 29, no. 1, pp. 84 – 106, 2013.

[7] GoNeighbour. (2010) Goneighbour. [Online]. Available: http://www.goneighbour.org/

[8] GoogleTM, "Google cloud," https://cloud.google.com/, 2016.

[9] D. Hughes, G. Coulson, and J. Walkerdine, "Free riding on gnutella revisited: The bell tolls?" *IEEE Distributed Systems Online*, vol. 6, no. 6, pp. 1–, Jun. 2005. [Online]. Available: http://dx.doi.org/10.1109/MDSO.2005.31

[10] A. L. Jin, W. Song, and W. Zhuang, "Auction-Based Resource Allocation for Sharing Cloudlets in Mobile Cloud Computing," *IEEE Transactions on Emerging Topics in Computing*, vol. PP, no. 99, pp. 1–1, 2015.

[11] S. A. Noor, R. Hasan, and M. M. Haque, "CellCloud: A Novel Cost Effective Formation of Mobile Cloud Based on Bidding Incentives," in *IEEE Int. Conf. on Cloud Computing (CLOUD)*, 2014, pp. 200–207.

[12] "Ns3 simulator," "https://www.nsnam.org/", 2015.

[13] P. Samimi, Y. Teimouri, and M. Mukhtar, "A combinatorial double auction resource allocation model in cloud computing," *Information Sciences*, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0020025514001054

[14] Z. Sun, Z. Zhu, L. Chen, H. Xu, and L. Huang, "A combinatorial double auction mechanism for cloud resource group-buying," in *IEEE Int. Performance Computing and Communications Conference (IPCCC)*, 2014, pp. 1–8.

[15] H. Wang, H. Tianfield, and Q. Mair, "Auction Based Resource Allocation in Cloud Computing," *Multiagent Grid Syst.*, vol. 10, no. 1, pp. 51–66, Jan. 2014. [Online]. Available: http://dx.doi.org/10.3233/MGS-140215

[16] K. Xu, Y. Zhang, X. Shi, H. Wang, Y. Wang, and M. Shen, "Online combinatorial double auction for mobile cloud computing markets," in *Performance Computing and Communications Conference (IPCCC), 2014 IEEE International*, Dec 2014, pp. 1–8.